DISH Second Course - Data Engineer Technical Assessment

Overview

This is a comprehensive 5-step technical assessment designed to evaluate your skills in data engineering, API integration, workflow orchestration, and containerization. The task follows a progressive difficulty approach, starting with simple data extraction and building up to advanced pipeline automation.

Submission: GitHub repository with complete solution or email the compressed file of your deliverables to `vineethshyam.kalidas@dish.digital` and `malte.meilenbrock@dish.digital`

Important Note:

This assessment is designed for learning and exploration, not perfection! We understand you have a full-time job and other commitments. Here's what we want you to know:

- **No perfect answers expected** - Show us your thought process and approach
- **One week timeframe** - Work at your own pace, focus on what you can accomplish
- **Learning opportunity** - Try new things, experiment, and document what you discover
- **Have fun with it** - This should be an engaging challenge, not a source of stress
- **Partial solutions welcome** - Better to do fewer steps well than rush through everything
- **Ask questions** - If you're stuck or need clarification, reach out to us

Remember: We're more interested in analytical thinking and approach than having every single requirement perfectly implemented.

Step 1: API Integration & Data Exploration

Objective

Familiarize yourself with the provided APIs and understand the data structures.

API Access Details

```
Base URL: https://dish-second-course-gateway-2tximoqc.nw.gateway.dev
API Key: AIzaSyDMMWBOHgMG1u7P9jX9neaUQHY2vwlBTbM
Endpoints:
```

```
/daily-visits (Beginner Level - Flat Data)
/ga-sessions-data (Advanced Level - Nested Data)

```

Available Date Ranges

- **Daily Visits**: 2016-08-01 to 2017-08-01 (YYYY-MM-DD format)
- **GA Sessions**: 2016-08-01 to 2017-08-01

Sample API Commands

Daily Visits (Simple Flat Data)

```
# Basic request
curl -H "X-API-Key: AIzaSyDMMWBOHgMG1u7P9jX9neaUQHY2vwlBTbM"
"https://dish-second-course-gateway-2tximoqc.nw.gateway.dev/daily-
visits?page=1&limit=10"

# With date filtering
curl -H "X-API-Key: AIzaSyDMMWBOHgMG1u7P9jX9neaUQHY2vwlBTbM"
"https://dish-second-course-gateway-2tximoqc.nw.gateway.dev/daily-
visits?start_date=2016-08-01&end_date=2016-08-05"
```

GA Sessions (Complex Nested Data)

```
# Basic request
curl -H "X-API-Key: AIzaSyDMMWBOHgMG1u7P9jX9neaUQHY2vwlBTbM"
"https://dish-second-course-gateway-2tximoqc.nw.gateway.dev/ga-
sessions-data?page=1&limit=5&date=20170801"

# With filters
curl -H "X-API-Key: AIzaSyDMMWBOHgMG1u7P9jX9neaUQHY2vwlBTbM"
"https://dish-second-course-gateway-2tximoqc.nw.gateway.dev/ga-
sessions-data?
date=20160801&country=United%20States&device_category=desktop&limit=3"
```

Sample Output Structures

Daily Visits Response (Flat Structure)

```
{
  "records": [
    {
      "visit_date": "2016-08-01",
      "total_visits": 1296
    },
    {
      "visit_date": "2016-08-02",
      "total_visits": 1963
    }
  ],
  "pagination": {
    "page": 1,
    "total_pages": 153,
    "has_next": true
  }
}
```

GA Sessions Response (Nested Structure)

```
{
  "records": [
    {
      "visitId": "1501615200123456789",
      "visitNumber": 1,
      "visitStartTime": 1501615200,

```

```
      "date": "20170801",
      "fullVisitorId": "0123456789098765432",
      "channelGrouping": "Organic Search",
      "totals": {
        "visits": 1,
        "hits": 5,
        "pageviews": 3,
        "bounces": null,
        "newVisits": 1
      },
      "device": {
        "browser": "Chrome",
        "operatingSystem": "Windows",
        "isMobile": false
      },
      "trafficSource": {
        "referralPath": null,
        "source": "google",
        "medium": "organic",
        "keyword": "analytics tutorial",
        "adContent": null
      },
      "geoNetwork": {
        "continent": "Americas",
        "country": "United States",
        "region": "California",
        "city": "Mountain View"
      },
      "customDimensions": [
        {"index": 1, "value": "logged_in"},
        {"index": 4, "value": "premium_user"}
      ],
      "hits_sample": [
        {
          "hitNumber": 1,
          "time": 0,
          "type": "PAGE",
          "isInteraction": true,
          "pagePath": "/analytics/tutorial",
          "pageTitle": "Google Analytics Tutorial",
          "hostname": "www.example.com"
        }
      ]
    }
  ]
}
```


Task requirements

Test both API endpoints using curl or Python requests

1. Explore the data structures and understand the differences
2. Document your findings in a README file

Step 2: Python module

Objective

Create a Python script to extract data from the API, flatten and load it into Google BigQuery.

2.1 Data Extraction

- Fetch data from both the API endpoint (daily-visits to start with)
- Implement pagination to retrieve all available data
- Handle API rate limits and errors gracefully

- Store raw API responses as JSON files in local storage location or Google Cloud Storage

2.2 Data Transformation

- **For Daily Visits**: Minimal transformation needed.
- **For GA Sessions**: Flatten nested structures into tabular format
- Handle null values appropriately
- Add metadata columns (load_timestamp, source_file(i.e. date), etc.)

2.3 Data Loading

- Load processed data into Google BigQuery
- Design an appropriate table schema based on the API response structure from Step 1
- Consider how to handle nested objects (flattened vs. JSON columns) and choose appropriate data types
- Include useful metadata columns for tracking and auditing
- **Bonus:** Explain your approach to handling duplicate data (upsert logic)
- **Bonus:** Describe what data quality checks you would implement

Expected Deliverables

- `data_pipeline.py` - Main extraction and loading script
- `requirements.txt` - Python dependencies
- `README.md` - Setup and execution instructions
- Sample output files in GCS or locally saved in zip file
- Populated BigQuery table or CSV equivalent

Step 3: Apache Airflow DAG Development

Objective

Convert your Python script into an Apache Airflow DAG with proper scheduling and monitoring.

3.1 DAG Configuration

- **Schedule**: Run at 6:00 AM and 6:00 PM on Wednesdays only
- **Retries**: 2 attempts with 5-minute delay
- **Timeout**: 3 minutes per task

3.2 Task Design

Design your DAG with appropriate tasks and dependencies to accomplish the ETL pipeline. Consider what steps are needed to Extract, Transform and Load data from both the API endpoints to BigQuery tables.

3.3 Bonus: Data Quality & Governance (Explanation Only)

Describe how you would implement data quality validation and governance practices:

Data Quality Monitoring:

- What metrics would you monitor (eg: null values, duplicates etc..)?
- How would you handle data quality failures?
- What alerting mechanisms would you implement for quality issues?

Data Governance & Compliance:

- How would you implement data lineage tracking for this pipeline?
- What metadata would you capture about data sources, transformations, and destinations?
- How would you ensure data privacy and handle sensitive information (PII)?
- What documentation and cataloging practices would you establish?

Expected Deliverables

- `etl_google_analytics_dag.py` - Complete Airflow DAG
- DAG documentation with task descriptions

Step 4: Docker Containerization

Objective

Create a simple Docker container for your Python data pipeline script.

4.1 Basic Containerization

- Create a Dockerfile that packages your Python data pipeline script
- Use a base image
- Install dependencies
- Copy your pipeline script and any configuration files
- Set an appropriate entry point to run your script

4.2 Configuration

- Use environment variables for API keys and GCP project settings
- Ensure the container can accept command-line arguments
- Document the required environment variables

Expected Deliverables

- `Dockerfile` - Basic container configuration
- `requirements.txt` - Python dependencies with versions
- `README.md` - Instructions on how to build and run the container
- Example docker run command with necessary environment variables

Step 5: AI Integration (Bonus)

Objective

Demonstrate your enthusiasm for AI by integrating a simple LLM-powered feature into your data pipeline.

5.1 Choose One Task

Pick one of the following beginner-friendly AI integrations:

Option A: Data Documentation Generator

- Use an LLM API (OpenAI, Google Gemini, etc.) to automatically generate documentation

- Input: Your BigQuery table schema and sample data
- Output: Human-readable data dictionary with column descriptions
- Example: "The 'visitStartTime' column contains Unix timestamps representing when a user session began"

Option B: Data Quality Report Generator

- Create a simple script that analyzes your loaded data
- Use an LLM to generate a natural language summary of data quality
- Input: Basic statistics (row counts, null percentages, data types)
- Output: Plain English report about data completeness and quality

Option C: SQL Query Assistant

- Build a simple tool that helps generate SQL queries for your data
- Use an LLM to convert natural language questions to SQL
- Example: "How many visits were there last week?" → Generate appropriate SQL query
- Test with 2-3 sample questions about your dataset

5.2 Implementation Guidelines

- Keep it simple - focus on learning and experimentation
- Use existing LLM APIs (don't build models from scratch)
- Include error handling for API calls
- Document what you learned about working with LLMs

Expected Deliverables

- Simple Python script with LLM integration (50-100 lines)
- Sample outputs showing the AI feature working
- Brief reflection (1 page) on what you learned about AI integration

Support

If you encounter any issues with the provided API or have questions about requirements, please reach out to:

Contact: vineethshyam.kalidas@dish.digital

Response Time: Within 24 hours on business days

Success Tips

1. **Start Simple**: Begin with the daily-visits endpoint before tackling GA sessions
2. **Incremental Development**: Get each step working before moving to the next
3. **Document Everything**: Clear documentation is as important as working code
4. **Test Thoroughly**: Validate your solution with different data ranges
5. **Think Production**: Consider scalability, monitoring, and maintenance

Good luck with your assessment!